Supplementary Information
CrossPlan: Systematic Planning of Genetic Crosses to Validate Mathematical Models

Aditya Pratapa, Neil Adames, Pavel Kraikivsky, Nicholas Franzese, John J. Tyson, Jean Peccoud, and T. M. Murali

S1 Extending CrossPlan to *Drosophila melanogaster*

To show how we can use CrossPlan for combinatorial genetics in other model organisms, we present an example of a series of crosses in *Drosophila* designed to obtain homozygous mutants in two loci starting with heterozygous single loss-of-function mutants. The homozygous single mutants both show a similar mild embryonic defect while the double mutant is expected to show synthetic embryonic lethality.

In the first cross (F0), the two parents are AABb and AaBB. The resulting offspring are AABB, AABb, AaBB, and AbBb, and all are viable. In the second cross (F1), the AbBb siblings are crossed to yield AABB, AABb, AaBB, AaBb, aaBB, aaBb, AAbb, Aabb, and aabb. If the two mutations result in embryonic lethality, we expect to see no adults with the aabb genotype. It is also possible that haploinsufficiency of one locus is sufficient for embryonic lethality when combined with the other locus as a homozygous null. In this case, we might obtain no offspring with the aabb and aaBb genotypes (b is haploinsuffient when a is homozygous), or aabb and Aabb genotypes (a is haploinsufficient when b is homozygous).

In summary, we can replace a single cross to obtain the necessary mutant in budding yeast with two crosses (F0 and F1) to get aabb from AABb and AaBB in *Drosophila*. With this change, the batch-based workflow and the CrossPlan algorithm are also useful for performing high-throughput genetics in diploid model organisms such as *Drosophila*.

S2 CrossPlan is NP-complete

In this section, we prove that the decision version of the CrossPlan is NP-complete. We first state this version of the problem.

DecideCrossPlan Problem. *Given a genetic cross graph $\mathcal{G} = (M, X, E)$, a set $S \subset M$ of source mutants, a set $T \subset M$ of target mutants, the batch size s, and the number of batches k, do k s-batches $\{W_i, 1 \leq i \leq k\}$ exist with the following properties?*

1. *For each $1 \leq i \leq k$, $In(W_i) \subseteq S \cup \left(\bigcup_{1 \leq j \leq i-1} Out(W_j)\right)$ and*
2. *$T \subseteq \left(\bigcup_{1 \leq j \leq k} Out(W_j)\right)$.*

The definitions of a batch and its input and output sets of mutants are the same as in the main text. In this decision version, we ask if there are k batches that can make all target mutants, i.e., T is a subset of the union of the outputs of all the batches.

We now state the Steiner tree problem, which we will reduce to the decision problem of the CrossPlan problem.

Steiner Tree Problem. *Given an undirected, weighted graph $G = (V, F)$, a set $U \subset V$ of terminal nodes, and an integer l, is there a connected subgraph of G with at most l edges that contains all the nodes in U?*

This problem is one of the 21 problems proved by Karp to be NP-complete (**?**). We will use $\langle G, U, l \rangle$ to denote such an instance of the Steiner tree problem.

Our reduction relies on establishing a correspondence between a traversal of a Steiner tree rooted at an arbitrary terminal in U and a sequence of crosses in an appropriately defined genetic cross graph. To effect this reduction, we construct an instance $\langle \mathcal{G}, S, T, s, k \rangle$ of DecideCrossPlan from $\langle G, U, k \rangle$, as described in Algorithm 1. In this algorithm, we use set notation to indicate which genes are deleted in a mutant instead of defining additional notation to name the mutant. We assume that the wild type (corresponding to the empty set) is viable.

Algorithm 1 Algorithm to construct an instance $\langle \mathcal{G}, S, T, s, k \rangle$ of DecideCrossPlan from an instance $\langle G, U, l \rangle$ of Steiner Tree.

1. Create a genetic cross graph $\mathcal{G} = (M, X, E)$ as follows:
   (a) For each node v in G, create two genes $g(v)$ and $d(v)$.
   (b) For each node v in G, create two viable mutant nodes $\{g(v)\}$ and $\{g(v), d(v)\}$ and one inviable mutant node $\{d(v)\}$. Add these mutant nodes to the set M.
   (c) For every undirected edge (u, v) in G, create five inviable mutant nodes corresponding to the gene sets $\{g(u), g(v)\}$, $\{g(u), d(v)\}$, $\{g(v), d(u)\}$, $\{g(u), g(v), d(v)\}$, and $\{g(v), g(u), d(u)\}$ in M.
   (d) For every undirected edge (u, v) in G, create two cross nodes in the set X as follows:
      (i) A cross node $x_{u,v}$ with incoming edges in E from the viable mutants $\{g(u)\}$ and $\{g(v), d(v)\}$ and outgoing edges in the edge set E to the eight mutants corresponding to the elements of the power set of $\{g(u), g(v), d(v)\}$.
      (ii) A cross node $x_{v,u}$ defined like $x_{u,v}$ but with the roles of u and v reversed.
2. Select an arbitrary node $u^* \in U$. Create the set S of source mutant nodes containing the mutant nodes $\{g(u^*)\}$ and all the mutant nodess of the form $\{g(v), d(v)\}$, where v is a node in G.
3. Create the set T of target mutant nodes containing all the mutant nodes of the form $\{g(u)\}$, where $u \in U$.
4. Set the batch size $s = 1$.
5. Set the number of batches $k = l$.

Figure S1 illustrates the crosses and the genetic cross graph constructed by this algorithm for a toy example.

We state a series of observations about the instance of DecideCrossPlan constructed by Algorithm 1.

Observation 1. *Every mutant node in M contains three or fewer gene deletions.*

Observation 2. *There are no mutant nodes in M of the form*

1. *$\{g(u), d(u), d(v)\}$, where (u, v) is an edge in G.*
2. *$\{g(u), d(v), d(w)\}$, where u, v, w are nodes in G.*
3. *$\{g(u), g(v), d(w)\}$, where u, v, w are nodes in G.*
4. *$\{g(u), g(v), g(w)\}$, where u, v, w are nodes in G.*
5. *$\{d(u), d(v), d(w)\}$, where u, v, w are nodes in G.*

Observation 3. *A mutant node of the form $\{g(u), g(v), d(v)\}$ is present in M iff (u, v) is an edge in G.*

Observation 4. *If $x_{u,v}$ is a cross node in X, then*

1. *both its input mutant nodes $\{g(u)\}$ and $\{g(v), d(v)\}$ are viable and*
2. *apart from $\{g(u)\}$ and $\{g(v), d(v)\}$ (which are also inputs to $x_{u.v}$) and the wild-type strain, the only other viable output mutant node of this cross is $\{g(v)\}$.*

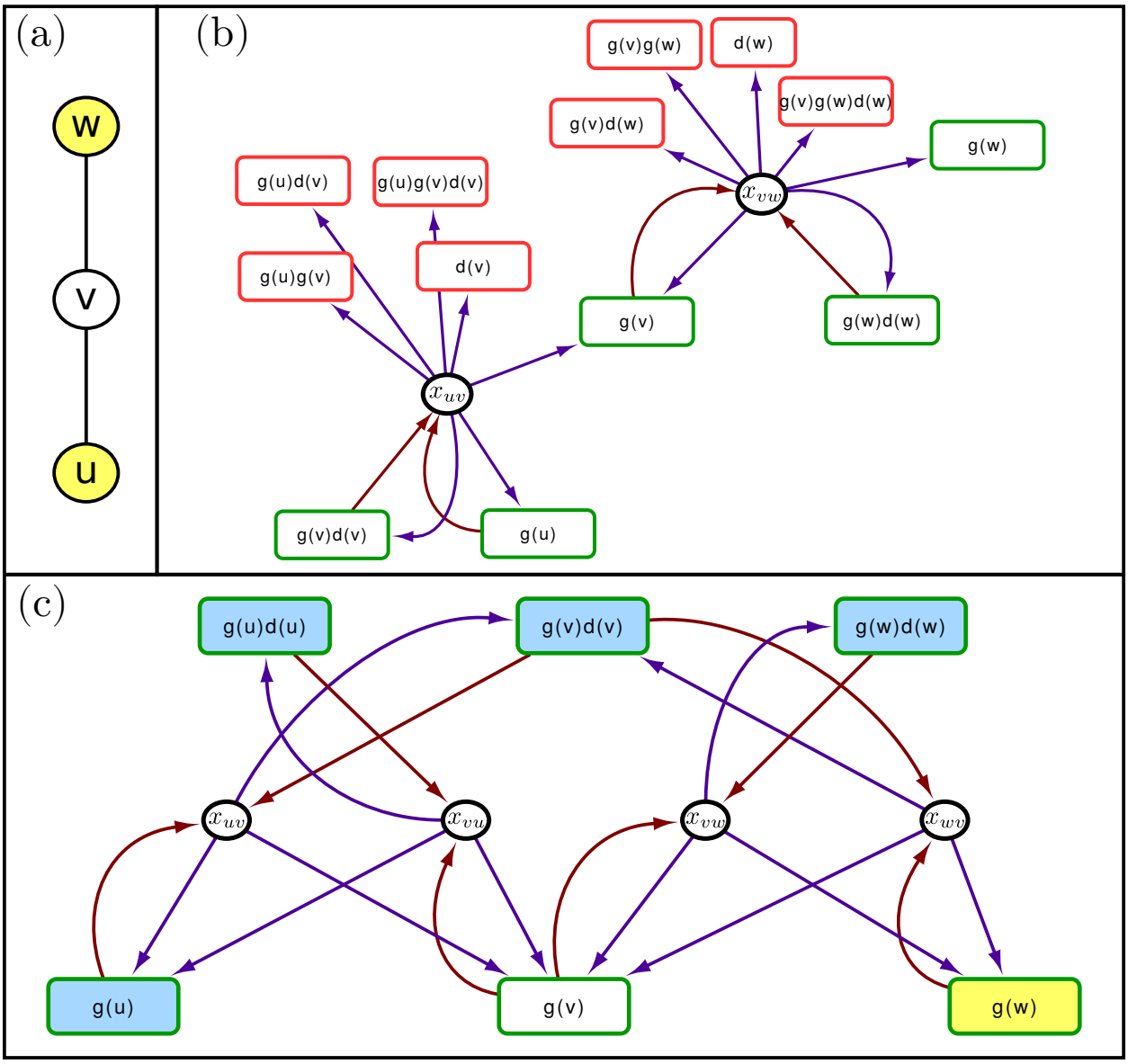


Figure S1: Illustration of cross nodes constructed by Algorithm 1. (a) Input graph to the Steiner tree problem. Yellow nodes denote terminals in U. (b) The cross nodes x_{uv} and x_{vw} constructed in Step 1d(i) Viable mutant nodes have green borders and inviable mutant nodes have red borders. Brown edges connect input mutant nodes to crosses and purple edges connect crosses to output mutant nodes. (c) The complete genetic cross graph constructed by the algorithm after the deletion of inviable mutant nodes. Blue rectangles correspond to mutant nodes in S and yellow rectangles to mutant nodes in T. In this example, u is the arbitrary node in G selected in Step 2. Note that the mutant node $\{g(u)\}$ is also a member of T but is not shown in yellow since it is a member of S.

Observation 5. *All mutant nodes in S and T are viable.*

Since no inviable mutant nodes are in T and inviable mutant nodes are not input into any cross nodes, we delete all inviable mutant nodes from $\mathcal{G}$ (Figure S1(c)). We can show that this modification will not affect any of the arguments we make below. However, the proof is tedious and we leave it to the reader. We can also state the following lemma.

Lemma 6. *The size of $\mathcal{G}$ is linear in the size of G. Algorithm 1 constructs the instance of DecideCrossPlan in time proportional to the size of G.*

We now prove a series of lemmas that show that if the instance of DecideCrossPlan created Algorithm 1 has a solution, then we can construct a solution to the corresponding Steiner tree problem. Recall that u^* is the arbitrary node in U selected in Step 2 of the algorithm.

Lemma 7. *If $\langle \mathcal{G}, S, T, s, k\rangle$ is a "Yes" instance of DecideCrossPlan, then there is a sequence $\langle x_1, x_2, \ldots, x_{k-1}, x_k\rangle$ of k cross nodes and a sequence $\langle v_1 = u^*, v_2, \ldots, v_k, v_{k+1}\rangle$ of $k+1$ mutant nodes in $\mathcal{G}$ such that for every $i, 1 \leq i \leq k$,*

1. *the inputs to cross node x_i are the mutant nodes $\{g(v_i)\}$ and $\{(g(v_{i+1}), d(v_{i+1})\}$*
2. *the output of x_i is the mutant node $\{g(v_{i+1})\}$, and*
3. *(v_i, v_{i+1}) is an edge in G.*

Proof. Since $s = 1$ (Step 4 in Algorithm 1), each batch in any solution to this instance of DecideCrossPlan contains one cross node. Let x_1, x_2, $\ldots x_k$ denote these cross nodes in batch order. By construction of $\mathcal{G}$ (Step 1d in Algorithm 1), each of these cross nodes corresponds to an edge of G. Therefore, we can rename the mutants involved in these cross nodes as follows: for each $1 \leq i \leq k$, the inputs to x_i are the mutant nodes $\{g(v_i)\}$ and $\{g(v_{i+1}), d(v_{i+1})\}$ and the output of x_i is the mutant node $\{g(v_{i+1})\}$. By construction, this cross node corresponds to the edge (v_i, v_{i+1}) in G.

The first cross node x_1 must have both input mutants in S. Only cross nodes that correspond to edges in G that are incident on u^* have this property. Hence, the node v_1 must be u^*, proving the lemma.

Note that it is possible that the sequence $\langle v_1 = u^*, v_2, \ldots, v_k, v_{k+1}\rangle$ does not contain $k+1$ distinct nodes, e.g., when there is an edge in G such that both crosses corresponding to it are elements of $\langle x_1, x_2, \ldots, x_{k-1}, x_k\rangle$ or when there is an index $j, 1 < j \leq k$, such that the output of x_j is identical to the input of an earlier cross. □

Let $\langle \mathcal{G}, S, T, s, k\rangle$ be a "Yes" instance of DecideCrossPlan. For each $1 \leq i \leq k$, we use G_i to denote the graph induced by the set of edges $\{(v_j, v_{j+1}), 1 \leq j \leq i\}$, where the edges are as defined in Lemma 7. The next two lemmas prove properties of these graphs.

Lemma 8. *If $\langle \mathcal{G}, S, T, s, k\rangle$ is a "Yes" instance of DecideCrossPlan, then for each $i, 1 \leq i \leq k$, the graph G_i is a connected subgraph of G containing u^*.*

Proof. Using the notation of Lemma 7, let the solution to this instance of DecideCrossPlan be the sequence of k cross nodes $\langle x_1, x_2, \ldots, x_{k-1}, x_k\rangle$. We prove the lemma by induction on i. The first cross node x_1 has $u^* = v_1$ as an input mutant node. Moreover G_1 contains one edge, proving the base case.

Assume that the lemma is true for the index $i > 1$, i.e., G_i is connected and contains u^*. We prove the statement for the index $i+1$. Consider the cross node x_{i+1}, which has the mutant nodes $\{g(v_i)\}$ and $\{g(v_{i+1}), d(v_{i+1})\}$ as inputs. Since the k crosses form a solution to this instance of DecideCrossPlan, each input must either be a member of S or the output of one of the crosses $x_1, x_2, \ldots, x_i$. Of the two inputs to x_{i+1}, only $\{g(v_{i+1}), d(v_{i+1})\}$ is an element of S, by construction (Step 2 of Algorithm 1). Hence, the mutant node $\{g(v_i)\}$ must be an output of a cross x_l, where $l \leq i$. Since G_i is connected, it must contain v_i. Therefore, G_{i+1}, which is the graph formed by the addition of the edge (v_i, v_{i+1}) to G_i, is also connected and contains u^*. Note that (v_i, v_{i+1}) may already be an edge in G_i or that the addition of this edge to G_i may create a cycle. □

Lemma 9. *If $\langle \mathcal{G}, S, T, s, k\rangle$ is a "Yes" instance of DecideCrossPlan, then the graph G_k is connected and contains all the terminals in U.*

Proof. Lemma 8 proves that G_k is connected and contains u^*. Since the crosses $x_1, x_2, \ldots, x_k$ correspond to a solution to the DecideCrossPlan problem, the outputs of these cross nodes must contain all the target mutants of the form $\{g_u\}$, where $u \in U - \{u^*\}$. In other words, for every terminal $u \in U - \{u^*\}$, there is at least one cross node $x_i, 1 \leq i \leq k$, such that the mutant node $\{g(u)\}$ is the output of x_i. Therefore, u is identical to v_{i+1}, i.e., u is a node in G_k, which proves the lemma. □

This lemma has the following corollary.

Corollary 10. *If $\langle \mathcal{G}, S, T, s, k\rangle$ is a "Yes" instance of DecideCrossPlan, then $\langle G, U, k\rangle$ is a "Yes" instance of Steiner tree, i.e., G_k contains a connected subgraph with at most k edges that contains all the nodes in U.*

Next, we consider how to construct a solution to the DecideCrossPlan problem from a "Yes" instance of Steiner tree.

Lemma 11. *If G contains a Steiner Tree with at most k edges that connects all the terminals in U, then the instance $\langle \mathcal{G}, S, T, s, k\rangle$ computed by Algorithm 1 is a "Yes" instance of DecideCrossPlan problem. Specifically, there are k cross nodes $\langle x_1, x_2, \ldots x_k\rangle$ in $\mathcal{G}$ such that*

1. *For each $1 \leq i \leq k$, $In(x_i) \subseteq S \cup \left(\bigcup_{1\leq j\leq i-1} Out(x_j)\right)$ and*
2. *$T \subseteq \left(\bigcup_{1\leq j\leq k} Out(x_j)\right)$.*

Proof. Let τ denote the Steiner tree in G with at most k edges that connects all the terminals in U. Consider a breadth-first traversal of τ starting at u^*. We rename the nodes in S as $v_1 = u^*$, v_2, …, v_{l+1} in the order of this traversal. We name the edges as follows: if an edge in S connects a node v_i to a node v_j, where $i < j$, we call this edge e_{j-1}. Note that edge indices lie between 1 and k and that the index of an edge is one less than the larger node index incident on that edge. We define a set of k cross nodes in $\mathcal{G}$ as follows: for each $1 \leq i \leq k$, let the edge e_i connect a vertex v_j (for some $j \leq i$) to the vertex v_{i+1}. Then the cross node x_i has the viable mutant nodes $\{g(v_j)\}$ and $\{g(v_{i+1}), d(v_{i+1})\}$ as inputs and the viable mutant node $\{g(v_{i+1})\}$ as output. Note that since e_i is an edge in G, this cross node and its input and output mutant nodes are members of $\mathcal{G}$ by construction (Steps 1b and 1d of Algorithm 1). We claim that this set of l cross nodes is a solution to the instance of DecideCrossPlan computed by Algorithm 1.

First, we prove by induction on i that for each $1 \leq i \leq k$, $In(x_i) \subseteq S \cup \left(\bigcup_{1\leq j\leq i-1} Out(x_j)\right)$. For $i = 1$, the inputs to x_1 are the mutant nodes $\{g(u^*)\}$ and $\{g(v_2), d(v_2)\}$, both of which are members of S by construction. Suppose the statement is true for index i. Consider the cross node x_{i+1}. Its inputs are the mutant nodes $\{g(v_j)\}$, for some $j \leq i$ and $\{g(v_{i+1}), d(v_{i+1})\}$. The second input is a member of S by construction. Since $j \leq i$ and $\{g(v_j)\}$ is the output of the cross x_j, we have shown that the statement holds true for index $i + 1$ as well.

Now we show that $T \subseteq \left(\bigcup_{1\leq j\leq k} Out(W_j)\right)$. Note that the right-hand side of this expression is simply the set of mutant nodes $\{\{g(v_j)\}, 1 \leq j \leq k\}$. Our construction of the crosses $x_1, x_2, \ldots, x_k$ from the Steiner tree τ implies that the set of nodes $\{v_k, 1 \leq j \leq k\}$ contain all the terminals in U. Therefore, T, which is the set of mutant nodes $\{\{g(u)\}, u \in U\}$ must be contained in $\{\{g(v_j)\}, 1 \leq j \leq k\}$. □

We are now ready to prove the main result of this section.

Theorem 12. *The DecideCrossPlan problem is NP-complete.*

Proof. We first prove that the DecideCrossPlan problem is in NP. Given a set of k batches $\{W_1, W_2, \ldots, W_k\}$, we perform the following checks for each $1 \leq i \leq k$. To perform these steps efficiently, we maintain a running set Ω of output mutant nodes, which we initialize with the mutants in S:

(i) We verify that every cross node in batch W_i is in $\mathcal{G}$ and that the size of W_i is at most s.

(ii) For every cross $x \in W_i$, for each of the two mutant nodes that is an input to x, we verify that the mutant node is an element of Ω.

(iii) For every cross $x \in W_i$, we add all the output mutant nodes of x to Ω.

(iv) If $i = k$, we check whether T is a subset of Ω.

The running time of this algorithm is polynomial in k, s, and the size of $\mathcal{G}$, proving that the DecideCrossPlan problem is in NP.

Consider the instance of DecideCrossPlan. Corollary 10 and Lemma 11 prove that the instance $\langle \mathcal{G}, S, T, s, k \rangle$ of DecideCrossPlan created by Algorithm 1 is a "Yes" instance of this problem iff $\langle G, U, l \rangle$ is a "Yes" instance of Steiner Tree. Since Steiner Tree is NP-complete, we have proven that DecideCrossPlan is also NP-complete. □

S3 Proof of Correctness for the CrossPlan ILP

In this section, we prove that the ILP described in the main text solves the CrossPlan problem correctly and optimally. For the sake of completeness, we first redefine the CrossPlan problem and restate the ILP used to solve it.

CrossPlan Problem. *Given a genetic cross graph $\mathcal{G} = (M, X, E)$, a set $S \subset M$ of source mutants, a set $T \subset M$ of target mutants, the batch size s, and the number of batches k, compute k s-batches $\{W_i, 1 \leq i \leq k\}$ such that*

1. *for each $1 \leq i \leq k$, $In(W_i) \subseteq S \cup \left(\bigcup_{1 \leq j \leq i-1} Out(W_j)\right)$ and*
2. *the size of $T \cap \left(\bigcup_{1 \leq j \leq k} Out(W_j)\right)$ is maximized over all possible sets of k s-batches.*

An ILP for CrossPlan. The ILP contains three sets of variables.

(i) For each cross node $x \in X$, we introduce k 0-1 variables $c_{x,i}$ where $1 \leq i \leq k$. We set $c_{x,i} = 1$ iff we perform that cross experiment in batch i.

(ii) For each mutant $m \in M$, we introduce $k+1$ 0-1 variables $b_{m,i}$ where $0 \leq i \leq k$. We set $b_{m,i} = 1$ iff we make mutant m in batch i. Over the course of several batches of experiments, we may make mutant m multiple times. Hence, $b_{m,i}$ may be set to 1 for multiple values of i.

(iii) For each mutant $m \in T$, we introduce one 0-1 variable a_m, which we set to 1 iff $b_{m,i} = 1$ for at least one value of i. We use the variables a_m to define the function we optimize below.

The ILP contains five sets of constraints.

(i) **Source mutants constraints**: Since the mutants in the source set S are already available, we can set their (and only their) b values in batch 0 to be unity:

$$b_{m,0} = \begin{cases} 1, & \text{for all } m \in S \\ 0, & \text{for all } m \notin S \end{cases} \tag{1}$$

(ii) **Batch size constraints**: We can perform at most s crosses in any batch.

$$\sum_{x \in X} c_{x,i} \leq s, \text{ for each } i, 1 \leq i \leq k \tag{2}$$

(iii) **Cross input constraints**: If we perform a genetic cross x in batch i, then we must have made both the mutants being crossed in x in one of the earlier batches.

$$c_{x,i} \leq \sum_{j=0}^{i-1} b_{m,j} \text{ for every } m \text{ such that } (m, x) \in E \text{ and for every } i, 1 \leq i \leq k \tag{3}$$

(iv) **Mutant input constraints**: If we make a mutant m in batch i, then we must also perform at least one of the genetic crosses that produces m in that batch.

$$b_{m,i} \leq \sum_{\substack{x \in X \\ (x,m) \in E}} c_{x,i}, \text{ for every } i, 1 \leq i \leq k \tag{4}$$

Note that (x, m) is an edge in the genetic cross graph iff m is one of the mutants that are the outputs of the cross x.

(v) **Making target mutant constraints**: For a target mutant m, $a_m = 1$ only if we make m in at least one batch.

$$a_m \leq \sum_{i=0}^{k} b_{m,i}, \text{ for each } m \in T \tag{5}$$

Our objective function is the following:

$$\max \sum_{m \in T} a_m \tag{6}$$

We seek to prove that an assignment of binary variables that is feasible, i.e., an assignment that satisfies constraints (1)–(5) and maximizes the objective function (6) is an optimal solution to the CrossPlan problem. We first consider the genetic cross graph corresponding to any feasible solution to this ILP and prove some of its properties. Let $\mathcal{O}$ be an assignment of the binary variables in the ILP in a feasible solution. We define the *output genetic cross graph* $\mathcal{G}(\mathcal{O}) = (M(\mathcal{O}), X(\mathcal{O}), E(\mathcal{O}))$ as follows (Figure S2(a),(b)):

1. For every mutant node m in $M(\mathcal{O})$, $b_{m,i} = 1$ for at least one value of $i, 0 \leq i \leq k$ in $\mathcal{O}$,
2. For every cross node x in $X(\mathcal{O})$, $c_{x,i} = 1$ in $\mathcal{O}$ for at least one value of $i, 1 \leq i \leq k$, and
3. $E(\mathcal{O})$ is the set of edges in the genetic cross graph $\mathcal{G}$ that are induced by the nodes in $M(\mathcal{O})$ and $X(\mathcal{O})$.

We need two more definitions. For every mutant $m \in M(\mathcal{O})$, we define $\beta(m)$ as the smallest batch index in which mutant m is made, i.e.,

$$\beta(m) = \arg\min_i \{b_{m,i} = 1, 0 \leq i \leq k\}.$$

For every cross $x \in X(\mathcal{O})$, we define $\beta(x)$ as the smallest batch index in which the cross is made, i.e.,

$$\beta(x) = \arg\min_i \{c_{x,i} = 1, 1 \leq i \leq k\}.$$

Note that these values are well-defined for every mutant node and cross node in $\mathcal{G}(\mathcal{O})$ since we include a mutant node m in $M(\mathcal{O})$ (respectively, cross node x in $X(\mathcal{O})$) iff $b_{m,i} = 1$ (respectively, $c_{x,i} = 1$) for at least one value of i. We can now state and prove the following lemma:

Lemma 13. *If $\mathcal{O}$ is a feasible solution to the ILP, then the output genetic cross graph $\mathcal{G}_{\mathcal{O}} = (M(\mathcal{O}), X(\mathcal{O}), E(\mathcal{O}))$ has the following properties:*

1. *Every cross node $x \in X(\mathcal{O})$ has in-degree equal to two.*
2. *If (m, x) is an edge in $E(\mathcal{O})$ from a mutant node m to a cross node x, then $\beta(m) < \beta(x)$.*

Proof. For every cross node $x \in X(\mathcal{O})$, the definition of $X(\mathcal{O})$ implies that $\beta(x)$ is defined; otherwise, we would not have included x in $X(\mathcal{O})$. Therefore, $c_{x,\beta(x)} = 1$. Similarly, if there is a mutant node m that is present in M but not in $M(\mathcal{O})$, then $b_{m,i} = 0$ for every $0 \leq i \leq k$; otherwise, we would have included m in $M(\mathcal{O})$.

1. Recall that $E(\mathcal{O})$ is the set of edges induced in $\mathcal{G}$ by the mutant nodes in $M(\mathcal{O})$ and the cross nodes in $X(\mathcal{O})$. Therefore, if there is a cross node $x \in X(\mathcal{O})$ whose in-degree is either zero or one, then at least one of the mutant nodes that are input to x in $\mathcal{G}$ is not present in $M(\mathcal{O})$. Let m be such a node. Note that (m, x) is an edge in $\mathcal{G}$. Since $\mathcal{O}$ is a feasible solution, constraint (3) must apply to this edge for $i = \beta(x)$, i.e., $c_{x,\beta(x)} \leq \sum_{j=0}^{\beta(x)-1} b_{m,j} = 0$, which contradicts the fact that $c_{x,\beta(x)} = 1$ by dint of the inclusion of x in $\mathcal{G}(\mathcal{O})$ (Figure S2(c)). Therefore, every cross node in $\mathcal{G}(\mathcal{O})$ must have in-degree equal to two.
2. Now suppose that $\beta(m) \geq \beta(x)$ for some $(m, x) \in E(\mathcal{O})$. Since constraint (3) must apply to this edge for $i = \beta(x)$, we have $1 = c_{x,\beta(x)} \leq \sum_{j=0}^{\beta(x)-1} b_{m,j}$. Therefore, $b_{m,j} = 1$ for at least one value of $0 \leq j \leq \beta(x) - 1$, which implies that $\beta(m) \leq \beta(x) - 1$, leading to a contradiction.

□

Now we define the batches corresponding to a feasible solution $\mathcal{O}$ and prove that each batch contains at most s crosses and that the batches satisfy the rules on input and output mutants in the statement of the CrossPlan problem. Given a feasible solution $\mathcal{O}$ to the ILP, for every $1 \leq i \leq k$, we define the *batch* $W_i(\mathcal{O})$ to be the set of crosses with $\beta(x) = i$, i.e., $W_i(\mathcal{O}) = \{x | c_{x,i} = 1, x \in X(\mathcal{O})\}$. Note that a cross may appear in multiple batches, by this definition. We can prove the following lemma.

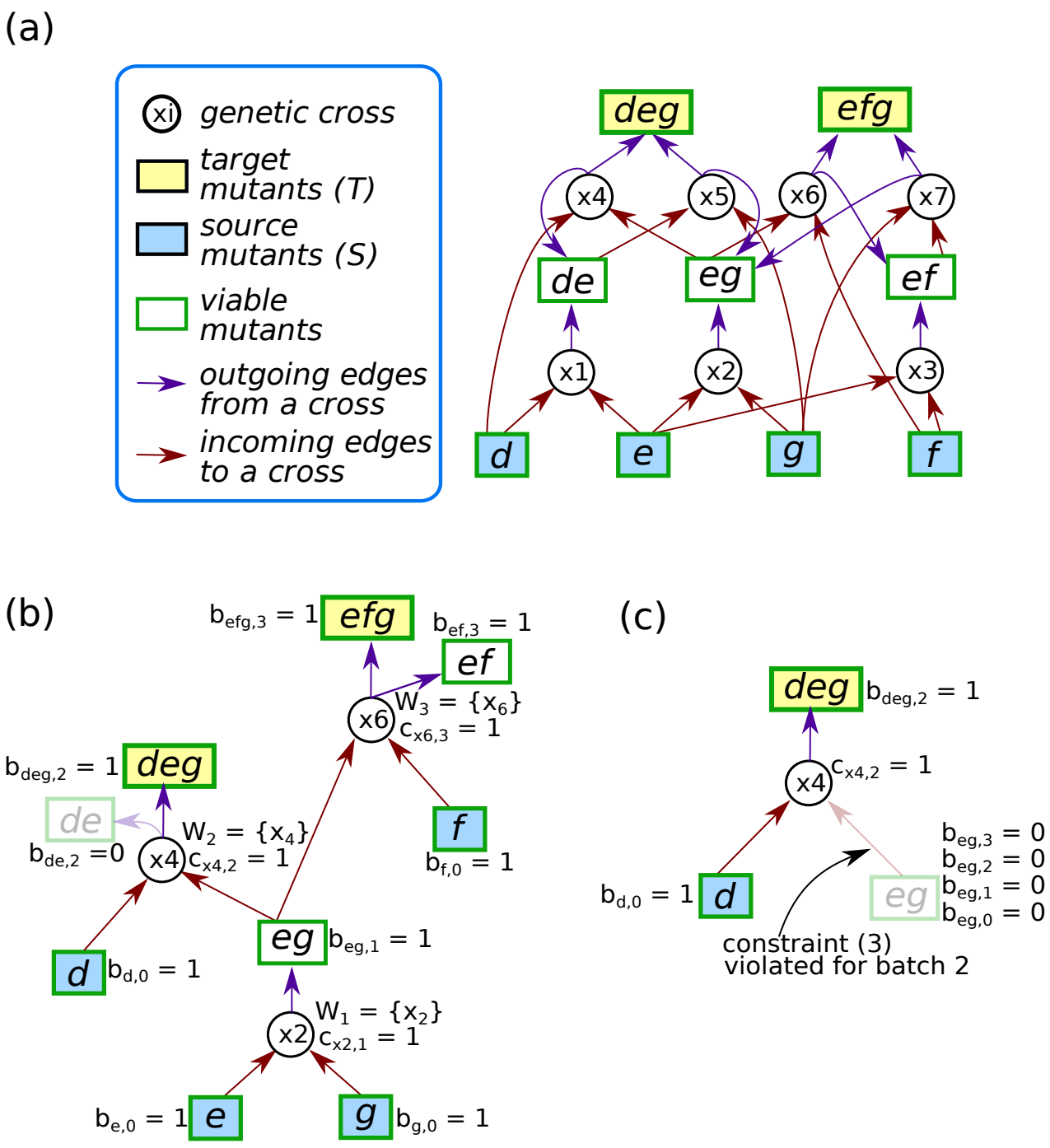


Figure S2: An illustration of a genetic cross graph and the output genetic cross graphs for a feasible solution (for $s = 1, k = 3$). (a) An input to the CrossPlan problem with four mutant nodes in S and two mutant nodes in T. We do not show inviable mutants. (b) The output genetic cross graph for a feasible solution where we perform cross $x2$ in batch 1, $x4$ in batch 2, and $x6$ in batch 3. For the mutant node de, the variable $b_{de,2} = 0$ in this feasible solution. Hence, this mutant node (displayed in a lighter color) is not an output of cross $x4$ in the genetic cross graph for this feasible solution. (c) Illustration of the proof of Lemma 13. If the mutant node eg is not in the output genetic cross graph, then constraint (3) is violated for the cross node $x4$ and batch 2.

Lemma 14. *For every $1 \leq i \leq k$, $W_i(\mathcal{O})$ contains at most s crosses.*

Proof. Consider an arbitrary value of i between 1 and k. Since $\mathcal{O}$ is a feasible solution to the ILP, it must satisfy constraint (2), i.e.,

$$\sum_{x \in X} c_{x,i} \leq s,$$

Therefore, the number of distinct cross nodes x such that $c_{x,i} = 1$ is at most s, which implies that the size of $W_i(\mathcal{O})$ is also at most s. □

With these lemmas in hand, we are almost ready to prove that the k s-batches $\{W_i(\mathcal{O}), 1 \leq i \leq k\}$ satisfy the rule on inputs and outputs in the statement of the CrossPlan problem. However, in the main text, we defined the inputs and outputs to a batch with respect to the input genetic cross graph $\mathcal{G}$ whereas we have defined batches for the feasible solution $\mathcal{O}$ with respect to the output genetic cross graph $\mathcal{G}(\mathcal{O})$, which is a subgraph of $\mathcal{G}$. To account for this difference, we first expand the notation of inputs and outputs to a batch: given a batch W, we will use $In(W, \mathcal{G})$ to represent the mutant nodes in $\mathcal{G}$ that are inputs to the cross nodes in W and $In(W, \mathcal{G}(\mathcal{O}))$ when we are referring to the mutant nodes in $\mathcal{G}(\mathcal{O})$. We use $Out(W, \mathcal{G})$ and $Out(W, \mathcal{G}(\mathcal{O}))$ analogously. We now state and prove some lemmas that relate the input and output sets of a batch in $\mathcal{G}$ and in $\mathcal{G}(\mathcal{O})$.

Lemma 15. *A mutant node m is in $\mathcal{G}(\mathcal{O})$ if and only if it is a member of S or is a member of the output set of some batch, i.e., $m \in S$ or $m \in Out(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O}))$ for some $i, 1 \leq i \leq k$.*

Proof. We first prove the "only if" direction. Consider a mutant node m that is in $\mathcal{G}(\mathcal{O})$. If $m \in S$, then this statement is proved in this direction. Therefore, we assume that $m \notin S$. Since m is in $\mathcal{G}(\mathcal{O})$, there must some value of $i, 1 \leq i \leq k$ such that $b_{m,i} = 1$. This variable must satisfy constraint (4), i.e.,

$$1 = b_{m,i} \leq \sum_{\substack{x \in X \\ (x,m) \in E}} c_{x,i}$$

Therefore, there must be a cross node x such that (x, m) is in $\mathcal{G}$ and $c_{x,i} = 1$. By construction, both x and (x, m) are in $\mathcal{G}(\mathcal{O})$ and x is an element of batch $W_i(\mathcal{O})$. Therefore, m is a member of $Out(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O}))$.

We now prove the "if" direction. There are two cases.

1. Case 1: Suppose m is in S. Constraint (1) sets $b_{m,0} = 1$. Therefore m is a node in $\mathcal{G}(\mathcal{O})$.
2. Case 2: Suppose m is an element of $Out(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O}))$ for some $i, 1 \leq i \leq k$. There must be a cross node x such that x and the edge (x, m) are in $\mathcal{G}(\mathcal{O})$; otherwise, m cannot be a member of $Out(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O}))$. Therefore m must also be a node in $\mathcal{G}(\mathcal{O})$.

□

Lemma 16. *Given a feasible solution $\mathcal{O}$ to the CrossPlan ILP, for each $i, 1 \leq i \leq k$, the s-batch $W_i(\mathcal{O})$ satisfies the following properties (Figure S3):*

1. *$In(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O})) = In(W_i(\mathcal{O}), \mathcal{G})$, i.e., the input mutants for $W_i(\mathcal{O})$ are identical in $\mathcal{G}(\mathcal{O})$ and in $\mathcal{G}$.*
2. *$Out(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O})) \subseteq Out(W_i(\mathcal{O}), \mathcal{G})$, i.e., the output mutants for $W_i(\mathcal{O})$ in $\mathcal{G}(\mathcal{O})$ are a subset of the output mutants in $\mathcal{G}$.*
3. *$In(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O})) \subseteq S \cup \left(\bigcup_{1 \leq j \leq i-1} Out(W_j(\mathcal{O}), \mathcal{G}(\mathcal{O})) \right)$, i.e., the input mutants for $W_i(\mathcal{O})$ in $\mathcal{G}(\mathcal{O})$ are a subset of the union of S and the output mutants of the preceding batches in $\mathcal{G}(\mathcal{O})$.*

Proof. We prove the claims in order.

1. Since $\mathcal{G}(\mathcal{O})$ is a subgraph of $\mathcal{G}$, we know that $In(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O})) \subseteq In(W_i(\mathcal{O}), \mathcal{G})$. We now prove that the two sets are identical. Consider any mutant node m that is a member of $In(W_i(\mathcal{O}), \mathcal{G})$. There must be a cross node x in $W_i(\mathcal{O})$ such that (m, x) is an edge in $\mathcal{G}$. By the definition of $W_i(\mathcal{O})$, x is a node in $\mathcal{G}(\mathcal{O})$ as well. By Lemma 13, the degree of x in $\mathcal{G}(\mathcal{O})$ is two. Therefore, m must be a member of both $M(\mathcal{O})$ and $In(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O}))$, proving the claim.

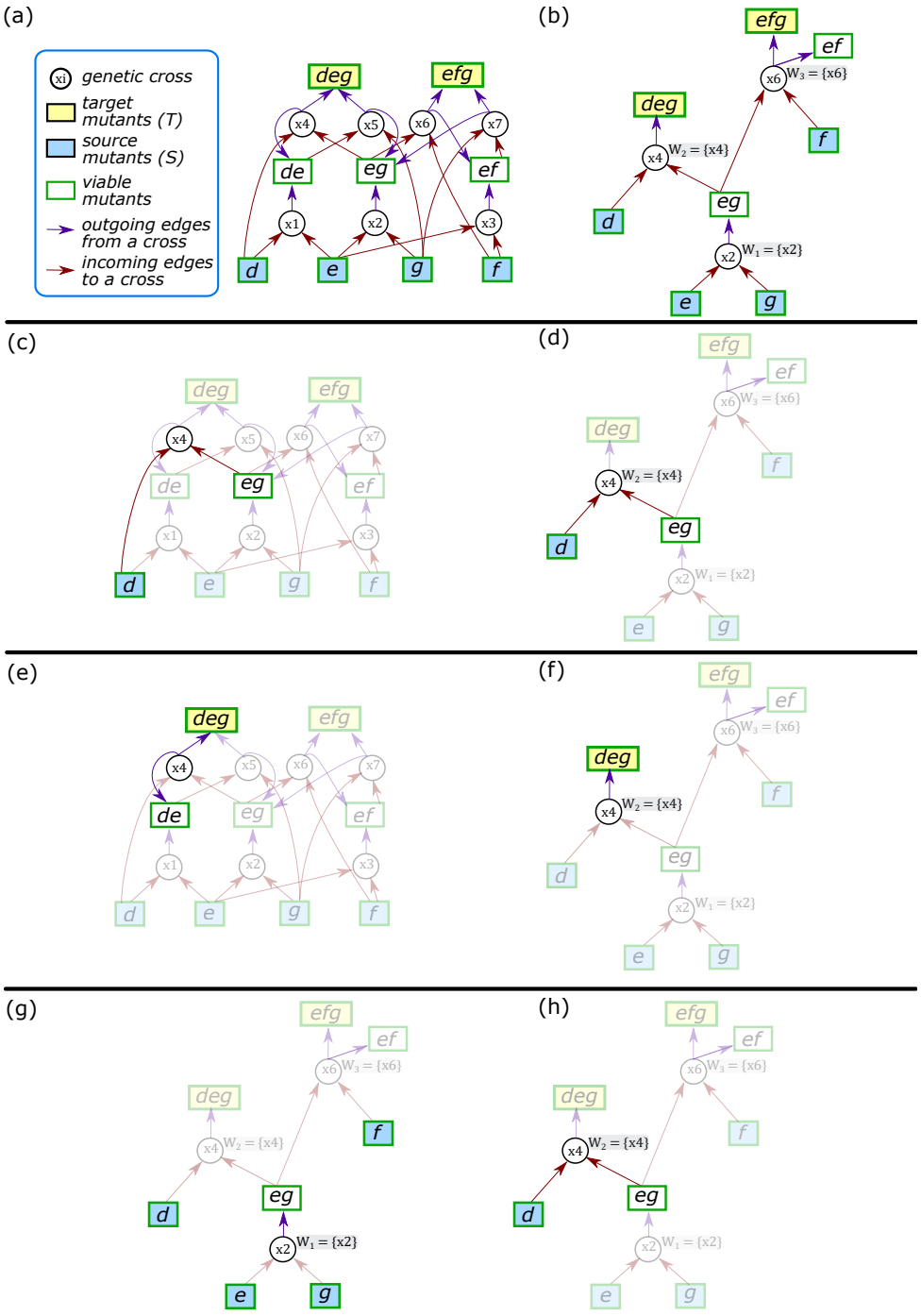


Figure S3: Illustration of the three statements in Lemma 16. (a) An input genetic cross graph $\mathcal{G}$ to the CrossPlan problem (identical to Figure S2(a)). (b) The output genetic cross graph $\mathcal{G}(\mathcal{O})$ for a feasible solution $\mathcal{O}$ (identical to Figure S2(b)). Here $s = 1, k = 3$. The remaining figures illustrate the statements in Lemma 16 for batch W_2, which contains the cross $x4$. (c) The input mutant nodes to W_2 in $\mathcal{G}$. (d) The input mutant nodes to W_2 in $\mathcal{G}(\mathcal{O})$. (e) The output mutant nodes of W_2 in $\mathcal{G}$. (f) The output mutant nodes of W_2 in $\mathcal{G}(\mathcal{O})$. (g) The union of S and the outputs of batch W_1. (h) The inputs to batch W_2, which are a subset of the mutant nodes in (g).

2. This statement follows from the fact that $\mathcal{G}(\mathcal{O})$ is a subgraph of $\mathcal{G}$.

3. Let m be an arbitrary mutant node in $In(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O}))$. By definition, there must be a cross node x in $W_i(\mathcal{O})$ such that (m, x) is an edge in $\mathcal{G}(\mathcal{O})$, i.e., m is an input mutant node to x in $\mathcal{G}(\mathcal{O})$. We know that for every $x \in W_i(\mathcal{O})$, we have $1 \leq \beta(x) \leq i$, and from Lemma 13, $\beta(m) < \beta(x) \leq i$. Consider batch $\beta(m)$. We have that $b_{m,\beta(m)} = 1$. There are two cases to consider.

   (a) $\beta(m) = 0$. In this case, m must be a member of S, which proves this statement in the Lemma.

   (b) $\beta(m) \geq 1$. In this case, we can apply constraint (4) for $i = \beta(m)$:
   $$1 = b_{m,\beta(m)} \leq \sum_{\substack{x \in X \\ (x,m) \in E}} c_{x,\beta(m)}.$$
   Therefore, there must be a cross node y such that m is an output of y and $c_{y,\beta(m)} = 1$, i.e., $y \in W_{\beta(m)}(\mathcal{O})$. Since $b_{m,\beta(m)} = 1$, $(y, m) \in \mathcal{G}(\mathcal{O})$, and $y \in W_{\beta(m)}(\mathcal{O})$, we have $m \in Out(W_{\beta(m)}(\mathcal{O}))$.

   Since $0 \leq \beta(m) < i$ and we chose an arbitrary mutant node m in $In(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O}))$, we have proven that $In(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O})) \subseteq S \cup \left(\bigcup_{1 \leq j \leq i-1} Out(W_j(\mathcal{O}), \mathcal{G}(\mathcal{O}))\right)$.

□

The following corollary is a consequence of Lemma 16. Note that we state this corollary in terms of $\mathcal{G}$.

Corollary 17. *Given a feasible solution $\mathcal{O}$ to the CrossPlan ILP, for each $1 \leq i \leq k$, the s-batch $W_i(\mathcal{O})$ satisfies $In(W_i(\mathcal{O}), \mathcal{G}) \subseteq S \cup \left(\bigcup_{1 \leq j \leq i-1} Out(W_j(\mathcal{O}), \mathcal{G})\right)$.*

We have now demonstrated that the s-batches defined by an feasible solution satisfy the first condition in the definition of the CrossPlan problem. We now state a lemma relating the expression in the objective function to the number of target mutants made by a feasible solution.

Lemma 18. *Given a feasible solution $\mathcal{O}$ to the CrossPlan ILP, the value of the objective function (6) is at most the number of target mutants that are also members of $\bigcup_{1 \leq j \leq k} Out(W_j(\mathcal{O}), \mathcal{G}(\mathcal{O}))$.*

Proof. Recall that the variable a_m is defined only if m is a target mutant. By constraint (5), if $a_m = 1$ then $\sum_{0 \leq i \leq k} b_{m,i} \geq 1$, i.e., $b_{m,i} = 1$ for at least one value of $i, 0 \leq i \leq k$. Therefore, by the proof of Lemma 15, m is a member of $Out(W_i(\mathcal{O}), \mathcal{G}(\mathcal{O}))$. However, it is possible that there is a target mutant m such that $b_{m,i} = 1$ for at least value of $0 \leq i \leq k$ but $a_m = 0$; this setting of variable values does not violate constraint (5). Therefore, the value of the objective function is at most the number of target mutants that are members of $\bigcup_{1 \leq j \leq k} Out(W_j(\mathcal{O}), \mathcal{G}(\mathcal{O}))$.

□

We now prove that maximizing the objective function (6) subject to the constraints (1)–(5) solves the CrossPlan problem to optimality.

Theorem 19. *Let $\mathcal{O}^*$ be a feasible solution to the CrossPlan ILP that maximizes the objective function (5). Then the k batches $\{W_i(\mathcal{O}^*), 1 \leq i \leq k\}$ satisfy the following properties:*

1. *For each $1 \leq i \leq k$, $|W_i(\mathcal{O}^*)| \leq s$, i.e., $W_i(\mathcal{O}^*)$ is an s-batch.*
2. *For each $1 \leq i \leq k$, $In(W_i(\mathcal{O}^*), \mathcal{G}) \subseteq S \cup \left(\bigcup_{1 \leq j \leq i-1} Out(W_j(\mathcal{O}^*), \mathcal{G})\right)$.*
3. *The size of $T \cap \left(\bigcup_{1 \leq j \leq k} Out(W_j(O^*), \mathcal{G})\right)$ is maximized over all possible sets of k s-batches.*

Proof. Since $\mathcal{O}^*$ is a feasible solution, the first two parts in the theorem follow from Lemma 14 and Corollary 17, respectively. We now prove the third statement in the theorem.

The value $\nu(\mathcal{O}^*)$ of the objective function for $\mathcal{O}^*$ satisfies the inequalities

$$\begin{aligned}
\nu(\mathcal{O}^*) &\leq T \cap \left(\bigcup_{1 \leq j \leq k} Out(W_j(\mathcal{O}^*), \mathcal{G}(\mathcal{O}^*)) \right), \text{ by Lemma 18} \\
&\leq T \cap \left(\bigcup_{1 \leq j \leq k} Out(W_j(\mathcal{O}^*), \mathcal{G}) \right), \text{ by Lemma 16}
\end{aligned}$$

Note that first inequality specifies the output sets of the batches with respect to $\mathcal{G}(\mathcal{O}^*)$ and the second inequality with respect to $\mathcal{G}$. We now prove that

$$\nu(\mathcal{O}^*) = T \cap \Big(\bigcup_{1 \leq j \leq k} Out(W_j(\mathcal{O}^*), \mathcal{G}) \Big).$$

If this equality does not hold, then there must be a target mutant m and a batch $1 \leq i \leq k$ and such that m is a member of $Out(W_i(\mathcal{O}^*), \mathcal{G})$ but $a_m = 0$. Since m is a member of $Out(W_i(\mathcal{O}^*), \mathcal{G})$, there must be a cross node x in $W_i(\mathcal{O}^*)$ such that (x, m) is an edge in $\mathcal{G}$. Clearly, x is in $\mathcal{G}(\mathcal{O}^*)$ (otherwise, it would not be a member of $W_i(\mathcal{O}^*)$). Since x in $W_i(\mathcal{O}^*)$, we have $c_{x,i} = 1$. Note that setting $b_{m,i} = 1$ does not violate constraint (4). We can now set $a_m = 1$ without violating constraint (5). However, we have constructed a new feasible solution to the ILP with an objective function value equal to $\nu(\mathcal{O}^*) + 1$, which contradicts the fact that $\mathcal{O}^*$ maximizes the objective function. Therefore, such a target mutant m cannot exists, which proves that $\nu(\mathcal{O}^*)$ equals the size of $T \cap \big(\bigcup_{1 \leq j \leq k} Out(W_j(\mathcal{O}^*), \mathcal{G}) \big)$. Since $\mathcal{O}^*$ maximizes the value of the objective function, we have also proven that the size of $T \cap \big(\bigcup_{1 \leq j \leq k} Out(W_j(O^*), \mathcal{G}) \big)$ is maximized over all possible sets of k s-batches. □

S4 Results for CrossPlan

S4.1 Analysis of CrossPlan output for $k = 12$

In this section, we discuss an extended analysis of the CrossPlan result for $k = 12$. We divided the crosses planned into two groups: crosses that resulted in at least one target mutant and crosses that produced only non-target mutants. Further, for each cross x, we recorded two batch indices: i_x, the batch in which the plan used x and j_x, the earliest batch in which x could have been used (i.e., the plan had created both mutants that were input to x in batch $j_x - 1$ or earlier). Note that $1 \leq j_x \leq i_x \leq k = 12$.

We observed that only 10 crosses yielded only non-target mutants (shown using black borders in Figure S4). Notably, we planned all these crosses in the very first batch, suggesting that the entire plan depended on finding an appropriate set of double gene mutants to make in batch one. In Figure S4, we also plot the value $i_x - j_x$ for each cross x made in batch i. The darker the color, the larger the value of $i_x - j_x$, i.e., the earlier we could have performed cross x. Figure S4 shows that many of the crosses planned in batches indices 9–12 could have been performed much earlier were it not for the restriction on the size of a batch.

S4.2 Analysis of mutants obtained using CrossPlan

The previous analysis suggests that many crosses could have been performed earlier if not for the restriction of the number of crosses per batch. Therefore, we investigated how many more target mutants we could make by merely crossing mutants obtained in the CrossPlan solution for k batches and without imposing any restriction on the number of crosses per batch. While there can be multiple optimal solutions to the CrossPlan problem, we based this analysis on one such solution that we computed for each value of k. Accordingly, we first solved CrossPlan for k batches where k ranged between 1 and 12. For each value of k, we then set S to be the set of mutants made in the computed plan for k batches, and solved another CrossPlan problem with $s = \infty$ (i.e., an unlimited batch size) and for one batch. By design, solving this new ILP will give us the maximum number of mutants that we can make just by crossing mutants obtained in the solution for k batches. We named the target mutants so obtained as "free target mutants", in the sense that these mutants are readily available by performing a single cross. We also computed the number of batches these crosses would require, assuming 12 crosses per batch, and called them "free batches".

Figures 5(a) and 5(b) summarize the results of this analysis. For $k = 1$, CrossPlan made 12 target mutants, which yielded only two free target mutants. All 14 target mutants were double mutants. The number of free target mutants increased substantially thereafter. For $k = 2$, where CrossPlan made 29 target mutants (Figure 4(a) in the main text), we could create 82 free target mutants, enough to occupy 7 free batches. The number of free target mutants increased linearly with k until $k = 9$. For $k = 9$, where CrossPlan made 161 target mutants (Figure 4(a) in the main text), we could create 327 free target mutants enough to occupy 28 batches. Thereafter, the number of free mutants hit a plateau. This analysis demonstrates that a viable approach to solving CrossPlan for k batches may be to solve it instead for a smaller

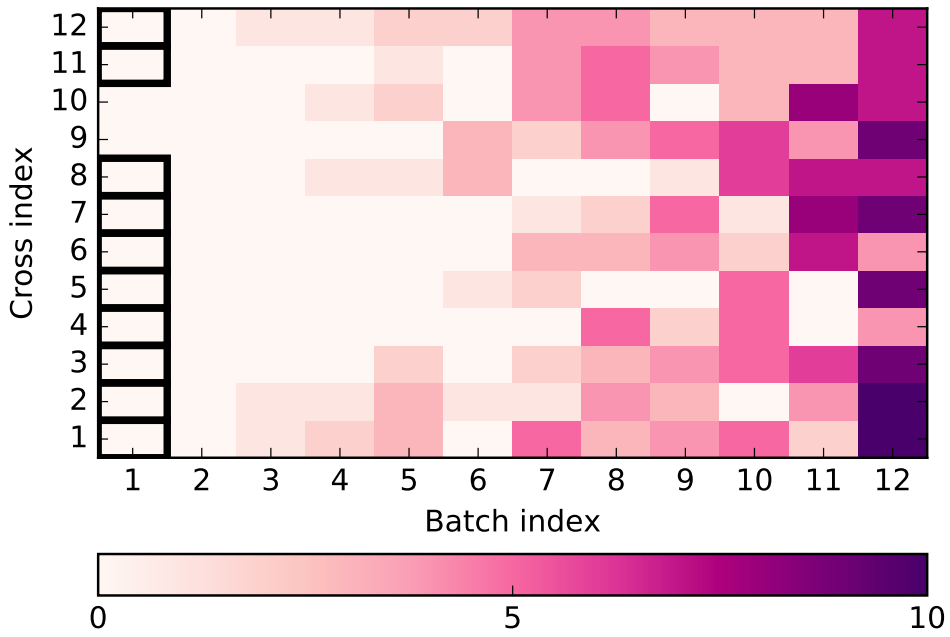
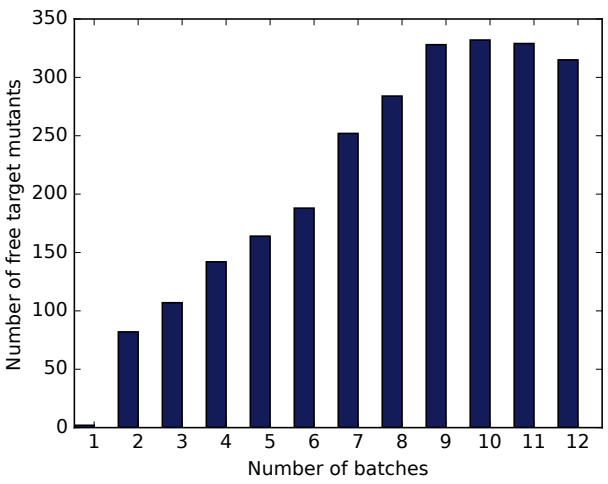
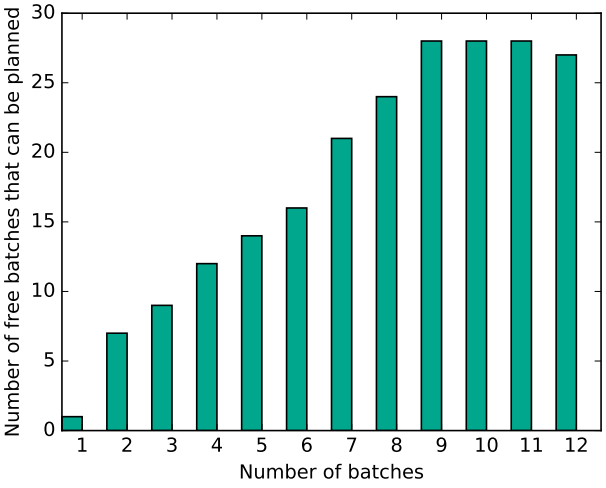


Figure S4: For the solution for $k = 12$, a heatmap showing the difference between the batch index in which CrossPlan performed a cross and the earliest batch in which it could have performed that cross. Each column corresponds to a batch in the solution for $k = 12$. Every cell in a column corresponds to a cross in that batch (ordered arbitrarily). As explained in the text, the darker the color in a cell, the earlier we could have performed that cross. Black borders denote crosses that yield only non-target mutants.

number k' of batches and then check if the number of free target mutants suffices to use the remaining $k - k'$ batches.



(a) Number of free target mutants.

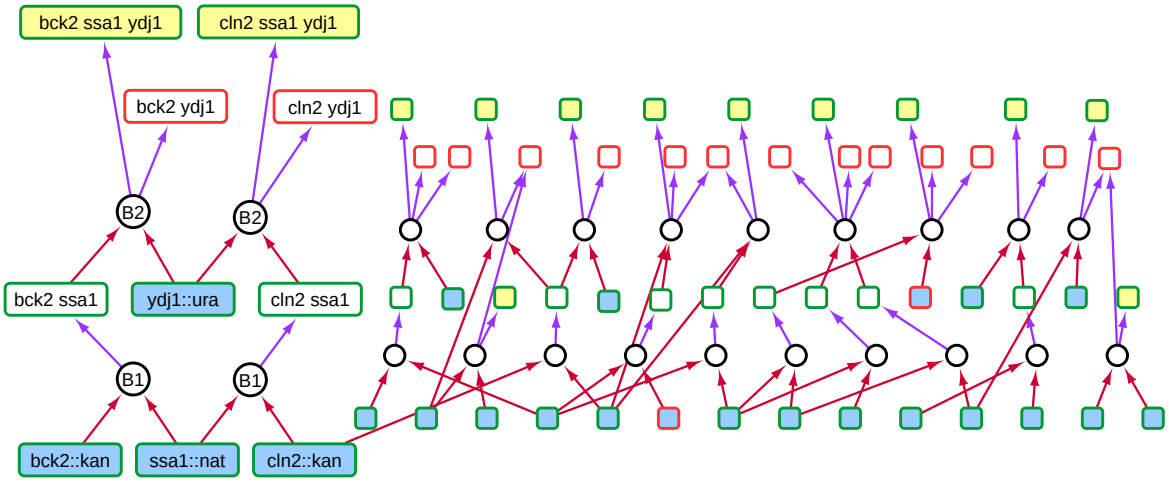


(b) Number of free batches required to make the free target mutants.

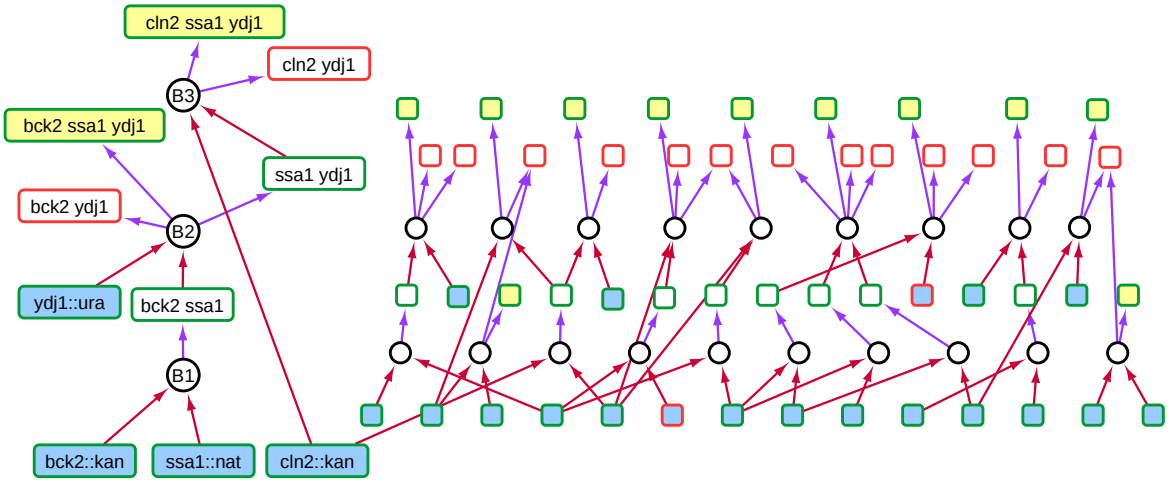
Figure S5: Analysis of target mutants that we can be make simply by crossing the mutants obtained in the CrossPlan solution for k batches, $1 \leq k \leq 12$.

S5 Comparison with Manual Planning

We manually searched for testable mutant rescue combinations that contained gene deletions known to alter the concentration of cell cycle activators/inhibitors. Moreover, these gene deletions should occur in some known and previously characterized double (or more) mutants that were used to parameterize the model. Therefore, we have high confidence in these model predictions, which are quite intuitive/logical, and include inviable combinations of cell cycle activators that could be counteracted by further removal of cell cycle inhibitors to rescue lethality. For example, Ssa1 and Ydj1 (Figure S6) have opposite effects on the G1 cyclin Cln3 (inhibitor and activator, respectively). The model predicts that *cln*2Δ *ydj*1Δ mutants are inviable because the activity of the G1 cyclins (Cln2 and Cln3) is too low to allow cell cycle progression from G1 to S phase. However, by removing Ssa1, Cln3 activity is elevated enough to support cell cycle progression. We found 13 such target mutants.



(a) Result from CrossPlan.

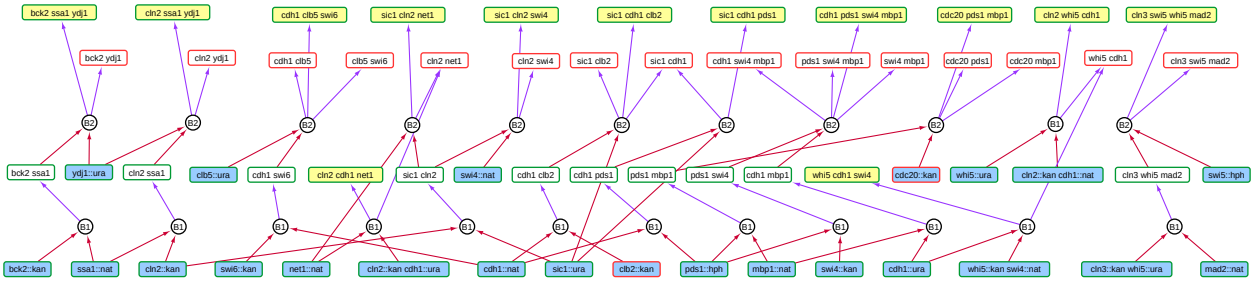


(b) Manually-created solution.

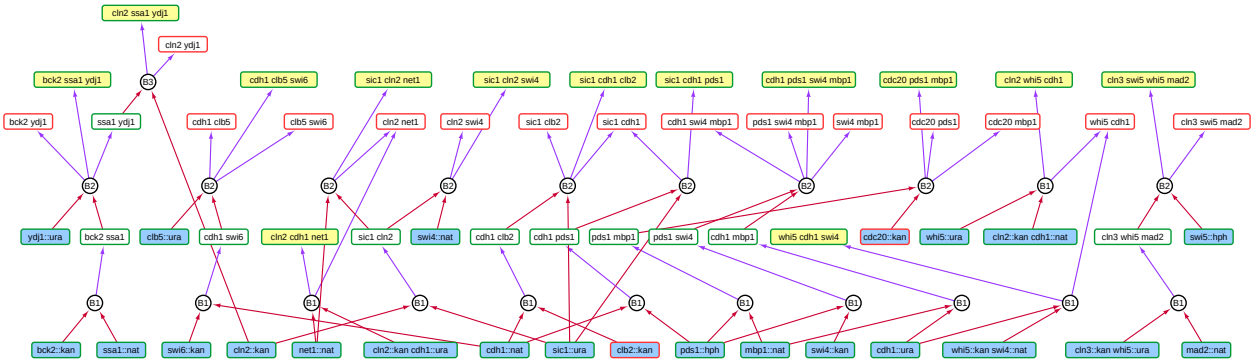
Figure S6: Comparison of CrossPlan's results and a manual solution for 13 target mutants. The terms "B1," "B2," and "B3" indicate that we plan a cross in batches one, two, and three, respectively. We only display a product of a cross if it is a target mutant (yellow with green border), an inviable parent of a target mutant (red border), or a viable mutant used in a subsequent cross. We indicate the marker for each gene in a starting mutant (blue).

Here, we study this set of 13 target mutants and compare two approaches to plan them: one computed by the ILP for CrossPlan and the other manually derived by one of the authors who is an expert yeast geneticist (Adames). For the set of starting mutants, we used 35 single gene mutants with four markers each, and three

different double mutants, of which one had two different markers sets. Figure 6(a) and 6(b) compare the two plans. In these figures, we show mutant names only where the plans differ and illustrate the full plans in fig. S7. We see that the CrossPlan ILP could plan all target mutants in two batches of twelve crosses each whereas the manual solution required three batches, with the third batch required to plan the twelfth target mutant. The key differences appear on the left-hand side of the figures, which depict how the two plans produce the mutant *cln*2Δ *ssa*1Δ *ydj*1Δ. To produce this mutant, CrossPlan crosses *ssa*1Δ with *cln*2Δ in the first batch and then crosses *cln*2Δ *ssa*1Δ with *ydj*1Δ in the second batch. In contrast, the manual solution does not produce any double mutant that is useful for *cln*2Δ *ssa*1Δ *ydj*1Δ in batch one. Instead, it crosses *ssa*1Δ *ydj*1Δ (from batch two) with *cln*2Δ to produce the desired mutant only in batch three. All the other crosses are identical. Another advantage of CrossPlan is that we could automatically compute a total of 17 different ways in which to plan these target mutants in two batches. Note that some other expert might create a plan identical to the one generated by CrossPlan. Our aim in this analysis was to show that there are clear advantages of using CrossPlan over manually planning many genetic cross experiments.



(a) Result from CrossPlan.



(b) Manually-created solution.

Figure S7: The complete genetic cross graphs corresponding to CrossPlan's result and a manual solution for 13 target mutants.